1.2 Introducing the Python Programming Language

For the thousands of years of human history before the mid-twentieth century, humans collected, analysed, and created data by hand. Digital computers were a revolution not just in technology but in civilization because of their ability to store more data than could fit on all the sheets of paper in the world, and to perform computations on this data faster and more reliably than an army of humans. Today, we rely on complex computer programs to operate on data in a variety of ways, from sending messages back and forth with loved ones, organizing data in documents and media, to running simulations of physical, social, and biological systems.

Yet for all their computation power, computers have one fundamental limitation: they have no agency, no inherent ability to make decisions about what to do. All they can do is take a set of (possibly very complex!) instructions, what we call a *computer program*, and execute them—no more, and no less. And so if we, as computer scientists, want to harness the awesome power of computers, we need to learn how give these instructions in a way that a computer understands. We need to learn how to speak to a computer.

What is a programming language?

A **programming language** is a way of communicating a set of instructions to a computer. Like human languages such as English, a programming language consists of a set of allowed words and the rules for putting those words together to form phrases with a coherent meaning.[1] Unlike human languages, a programming language must be precise enough to be understood by a computer, and so operates with a relatively small set of words and very structured rules for putting them together. Learning a programming language can be frustrating at first, because even a slight deviation from these rules results in the computer being unable to comprehend what we've written. But our time and efforts spent mastering the rules of a programming language yield a wonderful reward: the computer will not just understand what we're saying, but faithfully execute them.

[1] In your past learning of a (human) language, you've likely referred to these rules as the *grammar* of a language.

A **program** is simply the text of the instructions we wish to instruct the computer to execute; we call this text program **code** to differentiate it from other forms of text. To write programs in a particular language, we need to understand two key properties of the language. The first is the **syntax** of a programming language, which is the name we give to the rules governing what constitutes a valid program in the language. Before a computer can execute a program, it must read the instructions for the program; the syntax of the

programming language specifies the format of these instructions. The second concept is the **semantics** of a programming language, which refers to the rules governing the *meaning* of different instructions in the language. Once the computer has read the instructions in a program, it begins executing them. The language semantics specifies what the computer should do for each instruction.

The Python programming language

Just as there are thousands of human languages in the world today, each with their own vocabulary, grammar, and stylistic conventions, so too is there a plethora of programming languages that we can choose from. In this course, we'll use the Python programming language, which offers a simple, beginner-friendly syntax and a set of language instructions whose semantics are both powerful and accessible.

Now, neither our computer hardware nor operating system understand the Python programming language. Instead, the creators of the Python language developed a program called the **Python interpreter**, whose job is to take programs written in the Python language and execute the instructions.[2] You can think of the Python interpreter as a

[2] So when you "download Python", what you're actually downloading and installing is this Python interpreter software.

mediator between you the programmer, writing communicating in Python, and the computer hardware that actually executes instructions.

There are two ways of writing code in the Python language to be understood by the interpreter. The first is to write full programs in the Python language, saving them as text files,[3] and then running them through the Python interpreter. This is the standard way of

[3] Python programs use the `.py` file extension to distinguish them from other text files.

writing programs: write the instructions, and then run them with the interpreter. The second way is to run the Python interpreter in an interactive mode, which we call the **Python console** or **Python shell**. In this mode, we can write small fragments of Python code and ask the Python interpreter to execute each fragment one at a time. The Python console is useful for experimenting and exploring the language, as you get feedback after every single instruction. The drawback is that interactions with the interpreter in the Python console are ephemeral, lost every time you restart the console. So we'll use the following approach through the course: *use the Python console to learn about and experiment with the Python language, and write full programs in* `.py` *files*.

0:00 / 0:27

CSC110 Course Notes Home